12/5/1 (Item 1 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

06183619 E.I. No: EIP02447176713
Title: Data cache design considerations for the Itanium registered trademark processor
Author: Lyon, Terry; Delano, Eric; McNairy, Cameron; Mulla, Dean
Corporate Source: Hewlett-Packard, Fort Collins, CO, United States
Conference Title: International Conference on Computer Design (ICCD'02) VLSI in Copmuters and Processors
Conference Location: Freiburg, Germany Conference Date: 20020916-20020918
Sponsor: IEEE Computer Society
E.I. Conference No.: 60016
Source: Proceedings - IEEE International Conference on Computer Design: VLSI in Computers and Processors 2002. p 356-362
Publication Year: 2002
CODEN: PIIPE6
Language: English
Document Type: CA; (Conference Article) Treatment: G; (General Review)
Journal Announcement: 0211W1
Abstract: The second member in the Itanium Processor Family, the Itanium 2 processor, was designed to meet the challenge for high performance in today's technical and commercial server applications. The Itanium 2 processor's data **cache** microarchitecture provides abundant memory resources, low memory latencies and **cache** organizations tuned to for a variety of applications. The data **cache** design provides four memory ports to support the many performance optimizations available in the EPIC (Explicitly **Parallel** Instruction Computing) design concepts, such as prediction, speculation and explicit **prefetching** . The three-level **cache** hierarchy provides a 16KB 1-cycle **first level cache** to support the moderate bandwidths needed by integer applications. The **second level cache** is 256KB with a relatively low latency and FP balanced bandwidth to support technical applications. The on-chip third level **cache** is 3MB and is designed to provide the low latency and the large size needed by commercial and technical applications. 9 Refs.
Descriptors: Computer architecture; **Cache** memory; Storage allocation (computer); Parallel processing systems; Program compilers; Digital arithmetic; Response time (computer systems); Resource allocation; Computer simulation
Identifiers: Data **cache** design; Itanium processor family; Explicitly parallel instruction computing
Classification Codes:
722.1 (Data Storage, Equipment & Techniques); 722.4 (Digital Computers & Systems); 723.1 (Computer Programming); 721.1 (Computer Theory (Includes Formal Logic, Automata Theory, Switching Theory & Programming Theory)); 723.5 (Computer Applications)
722 (Computer Hardware); 723 (Computer Software, Data Handling & Applications); 721 (Computer Circuits & Logic Elements)
72 (COMPUTERS & DATA PROCESSING)

12/5/2 (Item 2 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

05942614 E.I. No: EIP01476735373
Title: Rank-preserving two-level caching for scalable search engines
Author: Saraiva, P.C.; De Moura, E.S.; Ziviani, N.; Meira, W.; Fonseca, R.; Ribeiro-Neto, B.
Corporate Source: Federal Univ. of Minas Gerais, Belo Horizonte, Brazil
Conference Title: 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval
Conference Location: New Orleans, LA, United States Conference Date: 20010909-20010913
Sponsor: IBM Research; AMERICA ONLINE; MICROSOFT RESEARCH; CIIR
E.I. Conference No.: 58701

12/5/1 (Item 1 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

06183619 E.I. No: EIP02447176713
Title: Data cache design considerations for the Itanium registered trademark processor
Author: Lyon, Terry; Delano, Eric; McNairy, Cameron; Mulla, Dean
Corporate Source: Hewlett-Packard, Fort Collins, CO, United States
Conference Title: International Conference on Computer Design (ICCD'02) VLSI in Copmuters and Processors
Conference Location: Freiburg, Germany Conference Date: 20020916-20020918
Sponsor: IEEE Computer Society
E.I. Conference No.: 60016
Source: Proceedings - IEEE International Conference on Computer Design: VLSI in Computers and Processors 2002. p 356-362
Publication Year: 2002
CODEN: PIIPE6
Language: English
Document Type: CA; (Conference Article) Treatment: G; (General Review)
Journal Announcement: 0211W1
Abstract: The second member in the Itanium Processor Family, the Itanium 2 processor, was designed to meet the challenge for high performance in today's technical and commercial server applications. The Itanium 2 processor's data **cache** microarchitecture provides abundant memory resources, low memory latencies and **cache** organizations tuned to for a variety of applications. The data **cache** design provides four memory ports to support the many performance optimizations available in the EPIC (Explicitly **Parallel** Instruction Computing) design concepts, such as prediction, speculation and explicit **prefetching** . The three-level **cache** hierarchy provides a 16KB 1-cycle **first level cache** to support the moderate bandwidths needed by integer applications. The **second level cache** is 256KB with a relatively low latency and FP balanced bandwidth to support technical applications. The on-chip third level **cache** is 3MB and is designed to provide the low latency and the large size needed by commercial and technical applications. 9 Refs.
Descriptors: Computer architecture; **Cache** memory; Storage allocation (computer); Parallel processing systems; Program compilers; Digital arithmetic; Response time (computer systems); Resource allocation; Computer simulation
Identifiers: Data **cache** design; Itanium processor family; Explicitly parallel instruction computing
Classification Codes:
722.1 (Data Storage, Equipment & Techniques); 722.4 (Digital Computers & Systems); 723.1 (Computer Programming); 721.1 (Computer Theory (Includes Formal Logic, Automata Theory, Switching Theory & Programming Theory)); 723.5 (Computer Applications)
722 (Computer Hardware); 723 (Computer Software, Data Handling & Applications); 721 (Computer Circuits & Logic Elements)
72 (COMPUTERS & DATA PROCESSING)

12/5/2 (Item 2 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

05942614 E.I. No: EIP01476735373
Title: Rank-preserving two-level caching for scalable search engines
Author: Saraiva, P.C.; De Moura, E.S.; Ziviani, N.; Meira, W.; Fonseca, R.; Ribeiro-Neto, B.
Corporate Source: Federal Univ. of Minas Gerais, Belo Horizonte, Brazil
Conference Title: 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval
Conference Location: New Orleans, LA, United States Conference Date: 20010909-20010913
Sponsor: IBM Research; AMERICA ONLINE; MICROSOFT RESEARCH; CIIR
E.I. Conference No.: 58701

Source: SIGIR Forum (ACM Special Interest Group on Information Retrieval) 2001. p 51-58
Publication Year: 2001
CODEN: FASRDV ISSN: 0163-5840
Language: English
Document Type: CA; (Conference Article) Treatment: T; (Theoretical); X; (Experimental)
Journal Announcement: 0111W4
Abstract: We present an effective **caching** scheme that reduces the computing and I/O requirements of a Web **search** engine without altering its ranking characteristics. The novelty is a **two - level caching** scheme that **simultaneously** combines **cached query** results and **cached** inverted lists on a real case **search** engine. A set of log queries are used to measure and compare the performance and the scalability of the search engine with no **cache** , with the **cache** for query results, with the **cache** for inverted lists, and with the two-level **cache** . Experimental results show that the two-level **cache** is superior, and that it allows increasing the maximum number of queries processed per second by a factor of three, while preserving the response time. These results are new, have not been reported before, and demonstrate the importance of advanced **caching** schemes for real case search engines. 32 Refs.
Descriptors: Search engines; **Cache** memory; Information retrieval systems; Query languages; Data processing; Websites
Identifiers: Two-level **caching**
Classification Codes:
723.1.1 (Computer Programming Languages)
903.3 (Information Retrieval & Use); 723.1 (Computer Programming); 723.2 (Data Processing)
723 (Computer Software, Data Handling & Applications); 903 (Information Science)
72 (COMPUTERS & DATA PROCESSING); 90 (ENGINEERING, GENERAL)

12/5/3 (Item 3 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

05192429 E.I. No: EIP98124511043
Title: Dynamic multithreading processor
Author: Akkary, Haitham; Driscoll, Michael A.
Corporate Source: Intel Corp
Conference Title: Proceedings of the 1998 31st Annual ACM/IEEE International Symposium on Microarchitecture
Conference Location: Dallas, TX, USA Conference Date: 19981130-19981202
Sponsor: IEEE
E.I. Conference No.: 49389
Source: Proceedings of the Annual International Symposium on Microarchitecture 1998. IEEE Comp Soc, Los Alamitos, CA, USA. p 226-236
Publication Year: 1998
CODEN: PSMIE7 ISSN: 1072-4451
Language: English
Document Type: CA; (Conference Article) Treatment: T; (Theoretical)
Journal Announcement: 9902W3
Abstract: We present an architecture that features dynamic multithreading execution of a single program. Threads are created automatically by hardware at procedure and loop boundaries and executed speculatively on a **simultaneous** multithreading pipeline. Data prediction is used to alleviate dependency constraints and enable **look** ahead execution of the threads. A **two - level** hierarchy significantly enlarges the instruction window. Efficient selective recovery from the second level instruction window takes place after a mispredicted input to a thread is corrected. The second level is slower to access but has the advantage of large storage capacity. We show several advantages of this architecture: (1) it minimizes the impact of ICache misses and branch mispredictions by fetching and dispatching instructions out-of-order, (2) it uses a novel value prediction and recovery mechanism to reduce artificial data dependencies created by the use of a stack to manage run-time storage, and (3) it improves the execution throughput of a superscalar by 15% without increasing the

execution resources or **cache** bandwidth, and by 30% with one additional ICache fetch port. The speedup was measured on the integer SPEC95 benchmarks, without any compiler support, using a detailed performance simulator. (Author abstract) 15 Refs.

Descriptors: *Reduced instruction set computing; Response time (computer systems); Microprogramming; Microprocessor chips; Constraint theory; Buffer storage; Storage allocation (computer); Bandwidth; Hierarchical systems; Data structures

Identifiers: Dynamic multithreading processors; Dispatching instructions

Classification Codes:

722.4 (Digital Computers & Systems); 723.1 (Computer Programming); 714.2 (Semiconductor Devices & Integrated Circuits); 721.1 (Computer Theory, Includes Formal Logic, Automata Theory, Switching Theory, Programming Theory); 722.1 (Data Storage, Equipment & Techniques)

722 (Computer Hardware); 723 (Computer Software); 714 (Electronic Components); 721 (Computer Circuits & Logic Elements)

72 (COMPUTERS & DATA PROCESSING); 71 (ELECTRONICS & COMMUNICATIONS)

12/5/4 (Item 4 from file: 8)

05016054 E.I. No: EIP98054209000

Title: Multilevel optimization of pipelined caches

Author: Olukotun, Kunle; Mudge, Trevor N.; Brown, Richard B.

Corporate Source: Stanford Univ, Stanford, CA, USA

Source: IEEE Transactions on Computers v 10 1997. p 1093-1102

Publication Year: 1997

CODEN: ITCOB4 ISSN: 0018-9340

Language: English

Document Type: JA; (Journal Article) Treatment: T; (Theoretical)

Journal Announcement: 9807W3

Abstract: The solution for solving problems of selecting the **cache** size and depth of **cache** pipelining that maximizes the performance of a given instruction-set architecture is formulated. The solution combines trace-driven architectural simulations and the timing analysis of the physical implementation of the **cache** . Increasing **cache** size tends to improve performance but this improvement is limited because **cache** access time increases with its size. This tradeoff results in an optimization problem **called** multilevel optimization, because it requires the **simultaneous** consideration of **two levels** of machine abstraction: the architectural level and the physical implementation level. 22 Refs.

Descriptors: *Buffer storage; Pipeline processing systems; Response time (computer systems); Computer architecture; Optimization; Problem solving; Computer simulation; Program processors; Data acquisition; Multichip modules

Identifiers: Multilevel optimization; Trace driven simulation; **Cache** pipelining

Classification Codes:

722.1 (Data Storage, Equipment & Techniques); 722.4 (Digital Computers & Systems); 921.5 (Optimization Techniques)

722 (Computer Hardware); 723 (Computer Software); 921 (Applied Mathematics)

72 (COMPUTERS & DATA PROCESSING); 92 (ENGINEERING MATHEMATICS)

12/5/5 (Item 5 from file: 8)

04594528 E.I. No: EIP97013493868

Title: Maximizing multiprocessor performance with the SUIF compiler

Author: Hall, Mary W.; Anderson, Jennifer M.; Amarasinghe, Saman P.; Murphy, Brian R.; Liao, Shih-Wei; Bugnion, Edouard; Lam, Monica S.

Source: Computer v 29 n 12 Dec 1996. p 84-89

Publication Year: 1996

CODEN: CPTRB4 ISSN: 0018-9162

Language: English
Document Type: JA; (Journal Article) Treatment: A; (Applications); G; (General Review)
Journal Announcement: 9703W1
Abstract: Multiple processors can work together to speed up single applications, but sequential programs must be rewritten to take advantage of the extra processors. One way to do this is through automatic parallelization with a compiler. Multiprocessors pose especially challenging problems for parallelizing compilers. Sufficient work must be performed in **parallel** to overcome processor **synchronization** and communication overhead. Moreover, multiprocessor memory hierarchies are complex, containing **both** shared memory and **multiple levels** of **cache** memory. Thus, two techniques are essential in **obtaining** good multiprocessor performance for array-based numerical programs: locating coarse-grain parallelism and managing multiprocessor memory use. The authors describe new technology in the Stanford SUIF compiler that enables it to successfully carry out these techniques. First, a suite of robust analysis techniques operate across procedure boundaries to locate coarse-grain parallelism so that large computations can execute independently in parallel. Then, to help eliminate **cache** misses, affine partitioning is used to improve processor reuse of data, and data permutation and data strip-mining make contiguous the data accessed by each processor in the shared address space. When employed in the automatic parallelizing compiler, these techniques significantly affect the performance of half the NAS and SPECfp95 benchmark suites. (Author abstract) 12 Refs.
Descriptors: *Parallel processing systems; Program compilers; Automation; Buffer storage; Storage allocation (computer); Computer aided analysis; Data acquisition
Identifiers: Shared address space; Data strip mining; Data permutations; Multiprocessor memory hierarchies
Classification Codes:
722.4 (Digital Computers & Systems); 723.1 (Computer Programming); 722.1 (Data Storage, Equipment & Techniques); 723.5 (Computer Applications); 723.2 (Data Processing)
722 (Computer Hardware); 723 (Computer Software); 731 (Automatic Control Principles)
72 (COMPUTERS & DATA PROCESSING); 73 (CONTROL ENGINEERING)

12/5/6 (Item 6 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

02720497 E.I. Monthly No: EI8903027950
Title: Multiprocessor system utilizing enhanced DSP's for image processing.
Author: Ueda, Hirotada; Kato, Kanji; Matsushima, Hitoshi; Kaneko, Kenji; Ejiri, Masakazu
Corporate Source: Hitachi Ltd, Tokyo, Jpn
Conference Title: Proceedings - International Conference on Systolic Arrays.
Conference Location: San Diego, CA, USA Conference Date: 19880525
Sponsor: IEEE, Computer Soc, Los Alamitos, CA, USA; US Office of Naval Research, USA; SPIE, USA
E.I. Conference No.: 11793
Source: Proc Int Conf on Systolic Arrays. Publ by IEEE, New York, NY, USA. Available from IEEE Service Cent (cat n 88CH2603-9) Piscataway, NJ, USA. p 611-620
Publication Year: 1988
ISBN: 0-8186-8860-2
Language: English
Document Type: PA; (Conference Paper) Treatment: X; (Experimental); ; A; (Applications)
Journal Announcement: 8903
Abstract: A general-purpose image processor (GPIP) consisting of 64 digital signal processors (DSPs) in a 0.31-m**3 box is proposed to perform a wide range of image processing tasks. A high-speed DSP **called** DSP-i has

been especially developed for this purpose. It has a highly **parallel** architecture with a **two - level** instruction hierarchy, multibank **cache** , and multiprocessor interface. The DSP-i machine cycle is 50 ns. A novel ring shift register bus architecture offers a flexible structure and an efficient data-exchange method for the system. Along with four proposed operation modes, it cuts the multiprocessing overhead to as little as 20%. The performance of the GPIP is 1000 MOPS (million operations per second). 5 Refs.

Descriptors: *SIGNAL PROCESSING--*Digital Techniques; COMPUTER ARCHITECTURE; IMAGE PROCESSING; COMPUTER SYSTEMS, DIGITAL--Parallel Processing

Identifiers: MULTIPROCESSOR SYSTEM; GENERAL PURPOSE IMAGE PROCESSOR (GPIP); DIGITAL SIGNAL PROCESSING (DSP)

Classification Codes:
716 (Radar, Radio & TV Electronic Equipment); 722 (Computer Hardware); 723 (Computer Software); 741 (Optics & Optical Devices)
71 (ELECTRONICS & COMMUNICATIONS); 72 (COMPUTERS & DATA PROCESSING); 74 (OPTICAL TECHNOLOGY)

12/5/7 (Item 1 from file: 35)

01791222 ORDER NO: AADAA-I9999435

Fuzzy discrete multicriteria cost optimization of steel structures using genetic algorithm

Author: Sarma, Kamal Chandra
Degree: Ph.D.
Year: 2001
Corporate Source/Institution: The Ohio State University (0168)
Adviser: Hojjat Adeli
Source: VOLUME 61/12-B OF DISSERTATION ABSTRACTS INTERNATIONAL.
PAGE 6614. 253 PAGES
Descriptors: ENGINEERING, CIVIL ; ENGINEERING, INDUSTRIAL
Descriptor Codes: 0543; 0546
ISBN: 0-493-08143-7

A great majority of optimization works on steel structures deal with minimum weight design. But in reality minimum weight design is not necessarily a minimum cost design. The price of commercially available rolled steel shapes in the market does not necessarily depend only on its weight but also on some other factors including demands and grades. In cost optimization, however, additional difficulties are encountered. They include the definition of cost function and uncertainties and fuzziness involved in determining the cost parameters. As a result only a small fraction of the structural optimization papers published deal with the minimization of the cost.

In this work a fuzzy discrete multicriteria initial cost optimization model has been developed by considering three criteria: (1) minimum cost, (2) minimum weight, and (3) minimum number of section types. This discrete cost optimization procedure is preceded by a fuzzy genetic continuous variable minimum weight design as a preliminary design. In this design the uncertainty or fuzziness of the AISC code based design constraints are considered. Furthermore, the fuzzy multicriteria initial cost optimization model is extended to perform a life cycle cost optimization of steel structures.

For optimization of large steel structures more than 99% of computation time is spent on the minimum weight design using the fuzzy Genetic Algorithm. Sequential processing of this optimization work in a single processor is very inefficient resulting in huge numbers of page faults and **cache** misses even in a supercomputer like Origin 2000. In this work **two** bi- **level** **parallel** processing algorithms are developed by using (1) data **parallel** procedure using OpenMP API at inner level and (2) message passing distributed parallel processing procedure by using function **calls** from MPI message passing library at outer level. Significant performance enhancement has been obtained in comparison to the traditional data parallel or distributed message passing parallel

processing.

12/5/8 (Item 1 from file: 2)
DIALOG(R)File 2:INSPEC

7502489 INSPEC Abstract Number: C2003-02-5310-005
Title: Data Cache design considerations for the Itanium/sub (R)/ 2 Processor
Author(s): Lyon, T.; Delano, E.; McNairy, C.; Mulla, D.
Author Affiliation: Hewlett-Packard Co., Fort Collins, CO, USA
Conference Title: Proceedings 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors p.356-62
Publisher: IEEE Comput. Soc, Los Alamitos, CA, USA
Publication Date: 2002 Country of Publication: USA xx+533 pp.
ISBN: 0 7695 1700 5 Material Identity Number: XX-2002-02537
U.S. Copyright Clearance Center Code: 1063-6404/02/$17.00
Conference Title: 2002 IEEE International Conference on Computer Design
Conference Sponsor: IEEE Comput. Soc. Tech. Committee on Design Autom.; IEEE Circuits & Syst. Soc
Conference Date: 16-18 Sept. 2002 Conference Location: Freiberg, Germany
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: The second member in the Itanium Processor Family, the Itanium 2 processor, was designed to meet the challenge for high performance in today's technical and commercial server applications. The Itanium 2 processor's data **cache** microarchitecture provides abundant memory resources, low memory latencies and **cache** organizations tuned to for a variety of applications. The data **cache** design provides four memory ports to support the many performance optimizations available in the EPIC (Explicitly **Parallel** Instruction Computing) design concepts, such as predication, speculation and explicit **prefetching** . The three-level **cache** hierarchy provides a 16KB 1-cycle **first level cache** to support the moderate bandwidths needed by integer applications. The **second level cache** is 256KB with a relatively low latency and FP balanced bandwidth to support technical applications. The onchip third level **cache** is 3MB and is designed to provide the low latency and the large size needed by commercial and technical applications. (7 Refs)
Subfile: C
Descriptors: **cache** storage; memory architecture; parallel architectures ; parallel machines; performance evaluation
Identifiers: Itanium 2 processor; data **cache** microarchitecture; memory resources; memory latencies; **cache** organizations; data **cache** ; performance optimizations; EPIC; Explicitly Parallel Instruction Computing; **cache** hierarchy; microarchitectures
Class Codes: C5310 (Storage system design); C5220P (Parallel architecture); C5470 (Performance evaluation and testing); C5440 (Multiprocessing systems)

12/5/9 (Item 2 from file: 2)
DIALOG(R)File 2:INSPEC

7387438 INSPEC Abstract Number: C2002-10-5340-003
Title: Speculative clustered caches for clustered processors
Author(s): Henry, D.S.; Loh, G.H.; Sami, R.
Author Affiliation: Dept. of Electr. Eng., Yale Univ., New Haven, CT, USA
Conference Title: High Performance Computing. 4th International Symposium, ISHPC 2002. Proceedings (Lecture Notes in Computer Science Vol.2327) p.281-90
Editor(s): Zima, H.P.; Joe, K.; Sato, M.; Seo, Y.; Shimasaki, M.
Publisher: Springer-Verlag, Berlin, Germany
Publication Date: 2002 Country of Publication: Germany xv+564 pp.
ISBN: 3 540 43674 X Material Identity Number: XX-2002-01484

Conference Title: High Performance Computing. 4th International Symposium, ISHPC 2002. Proceedings
Conference Date: 15-17 May 2002 Conference Location: Kansai Science City, Japan
Language: English Document Type: Conference Paper (PA)
Treatment: Applications (A); Practical (P)
Abstract: Clustering is a technique for partitioning superscalar processor's execution resources to simultaneously allow for more in-flight instructions, wider issue width, and more aggressive clock speeds. As either the size of individual clusters or the total number of clusters increases, the distance to the first level data **cache** increases as well. Although clustering may expose more parallelism by allowing a greater number of instructions to be **simultaneously** analyzed and issued, the gains may be obliterated if the latencies to memory grow too large. We propose to augment each cluster with a small, fast, simple **Level Zero** (**L0**) data **cache** that is accessed in **parallel** with a traditional **L1** data **cache** . The difference between our solution and other proposed **caching** techniques for clustered processors is that we do not support versioning or coherence. This may occasionally result in a load instruction that **reads** a stale value from the L0 **cache** , but the common case is a low latency hit in the L0 **cache** . Our simulation studies show that 4 KB, 2-way set associative L0 **caches** provide a 6.5-12.3% IPC improvement over a wide range of processor configurations. (19 Refs)
Subfile: C
Descriptors: **cache** storage; content-addressable storage; parallel processing
Identifiers: speculative clustered **caches** ; clustered processors; in-flight instructions; aggressive clock speeds; data **cache** ; level zero data **cache** ; **caching** techniques
Class Codes: C5340 (Associative storage); C1230D (Neural nets); C6120 (File organisation); C5440 (Multiprocessing systems)

12/5/10 (Item 3 from file: 2)
DIALOG(R)File 2:INSPEC

6933725 INSPEC Abstract Number: C2001-07-6120-003
Title: Matrix multiplication: a case study of enhanced data cache **utilization**
Author(s): Eiron, N.; Rodeh, M.; Steinwarts, I.
Author Affiliation: Dept. of Comput. Sci., Technion-Israel Inst. of Technol., Haifa, Israel
Journal: ACM Journal of Experimental Algorithmics Conference Title: ACM J. Exper. Algorithmics (USA) vol.4
Publication URL: http://www.jea.acm.org/
Publisher: ACM,
Publication Date: 1999 Country of Publication: USA
ISSN: 1084-6654
Material Identity Number: E335-2001-004
Conference Title: 2nd Workshop on Algorithm Engineering. WAE'98
Conference Date: 20-22 Aug. 1998 Conference Location: Saarbrucken, Germany
Language: English Document Type: Conference Paper (PA); Journal Paper (JP)
Treatment: Practical (P)
Abstract: Modern machines present two challenges to algorithm engineers and compiler writers: they have superscalar, super-pipelined structure, and they have elaborate memory subsystems specifically designed to reduce latency and increase bandwidth. Matrix multiplication is a classical benchmark for experimenting with techniques used to exploit machine architecture and to overcome the limitations of contemporary memory subsystems. This research aims at advancing the state of the art of algorithm engineering by balancing instruction level parallelism, **two levels** of data tiling, copying to provably avoid any **cache** conflicts, and **prefetching** in **parallel** to computational operations, in order to fully exploit the memory bandwidth. Measurements on IBM's RS/6000 43P

workstation show that the resultant matrix multiplication algorithm outperforms IBM's ESSL by 6.8-31.8%, is less sensitive to the size of the input data, and scales better. We introduce a **cache** aware algorithm for matrix multiplication. We also suggest generic guidelines that may be applied to compute intensive algorithm to efficiently utilize the data **cache** . We believe that some of our concepts may be embodied in compilers. (17 Refs)

Subfile: C

Descriptors: **cache** storage; IBM computers; mathematics computing; matrix multiplication; parallel programming; program compilers

Identifiers: matrix multiplication; data **cache** utilization; superscalar super-pipelined structure; memory subsystems; instruction level parallelism ; data tiling; prefetching; IBM RS/6000 43P workstation; **cache** aware algorithm; program compilers

Class Codes: C6120 (File organisation); C4140 (Linear algebra (numerical analysis)); C7310 (Mathematics computing); C6150N (Distributed systems software); C6150C (Compilers, interpreters and other processors)

12/5/11 (Item 4 from file: 2)

DIALOG(R)File 2:INSPEC

6103923 INSPEC Abstract Number: C9901-4140-016

Title: Matrix multiplication: a case study of algorithm engineering

Author(s): Eiron, N.; Rodeh, M.; Steinwarts, I.

Author Affiliation: Dept. of Comput. Sci., Technion-Israel Inst. of Technol., Haifa, Israel

Conference Title: WAE'98. 2nd Workshop on Algorithm Engineering. Proceedings p.98-109

Editor(s): Mehlhorn, K.

Publisher: Max-Planck-Inst. Inf, Saarbrucken, Germany

Publication Date: 1998 Country of Publication: Germany iv+213 pp.

Material Identity Number: XX98-02236

Conference Title: 2nd Workshop on Algorithm Engineering. WAE'98

Conference Date: 20-22 Aug. 1998 Conference Location: Saarbrucken, Germany

Language: English Document Type: Conference Paper (PA)

Treatment: Practical (P)

Abstract: Modern machines present two challenges to algorithm engineers and compiler writers: they have superscalar, super-pipelined structure, and they have elaborate memory subsystems specifically designed to reduce latency and increase bandwidth. Matrix multiplication is a classical benchmark for experimenting with techniques used to exploit machine architecture and to overcome the limitations of contemporary memory subsystems. This research aims at advancing the state of the art of algorithm engineering by balancing instruction level parallelism, **two levels** of data tiling, copying to provably avoid any **cache** conflicts, and **prefetching** in **parallel** to algorithmic operations, in order to fully exploit the memory bandwidth. Measurements show that the resultant matrix multiplication algorithm outperforms IBM's ESSL by 6.8-31.8%, is less sensitive to the size of the input data, and scales better. The techniques presented in the paper have been developed specifically for matrix multiplication. However, they are quite general and may be applied to other numeric algorithms. We believe that some of our concepts may be generalized to be used as compile-time techniques. (17 Refs)

Subfile: C

Descriptors: **cache** storage; floating point arithmetic; matrix multiplication; parallel algorithms; parallelising compilers; software performance evaluation

Identifiers: matrix multiplication; algorithm engineering; superscalar super-pipelined structure; elaborate memory subsystems; machine architecture; instruction level parallelism; data tiling; copying; compile-time techniques

Class Codes: C4140 (Linear algebra (numerical analysis)); C6150C (Compilers, interpreters and other processors); C6110P (Parallel programming); C5230 (Digital arithmetic methods)

12/5/12 (Item 5 from file: 2)
DIALOG(R)File 2:INSPEC

5816469 INSPEC Abstract Number: C9803-5220P-009
Title: The architecture of OCMP and its evaluation
Author(s): Saisho, K.; Sano, T.; Fukuda, A.
Author Affiliation: Graduate Sch. of Inf. Sci., Nara Inst. of Sci. & Technol., Takayama, Japan
Conference Title: Proceedings. Third International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN '97) (Cat. No.97TB100209) p.71-7
Editor(s): Lai, F.; Maggs, B.; Hsu, D.F.
Publisher: IEEE Comput. Soc, Los Alamitos, CA, USA
Publication Date: 1997 Country of Publication: USA xiii+503 pp.
ISBN: 0 8186 8259 6 Material Identity Number: XX98-00002
U.S. Copyright Clearance Center Code: 1087-4087/97/$10.00
Conference Title: Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN'97)
Conference Sponsor: Nat. Taiwan Univ.; Minstr. Educ.; Nat. Sci. Council ROC; Inst. Inf. & Comput. Machinery (IICM); ACM Taipei/Taiwan Chapter; IEICE Inf. Syst. Soc. Japan; Inf. Process. Soc. Japan
Conference Date: 18-20 Dec. 1997 Conference Location: Taipei, Taiwan
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: By gathering multiple processors in one LSI chip, communication delay between processors becomes shorter and then efficient fine/medium grain **parallel** processing can be realized. The authors propose a new processor architecture **called** OCMP (On-Chip Multi-Processing Architecture). OCMP has **two** characteristics: one is the instruction **level** dispatching mechanism; and the other is the divided **cache** system. OCMP employs a fork-join type **parallel** processing model in order to simplify the dispatching mechanism. By dividing the **cache** system into shared **cache** and private **cache** , the **cache** coherence problem between processors on the same chip is removed and access conflict on the shared **cache** is also relaxed. OCMP is evaluated with the instruction level simulator developed by the authors. Two types of instruction level dispatching mechanisms are compared. The memory access mechanism is evaluated with various parameters such as memory access cost, the degree of simultaneous access to shared **cache** , and so on. (9 Refs)
Subfile: C
Descriptors: **cache** storage; instruction sets; large scale integration; multiprocessing systems; multiprocessor interconnection networks; parallel architectures; performance evaluation
Identifiers: OCMP architecture; multiple processors; LSI chip; communication delay; fine grain parallel processing; medium grain parallel processing; processor architecture; On-Chip Multiprocessing Architecture; instruction level dispatching; divided **cache** system; fork-join type parallel processing; shared **cache** ; private **cache** ; **cache** coherence; access conflict; instruction level simulator; memory access
Class Codes: C5220P (Parallel architecture); C5440 (Multiprocessing systems); C5470 (Performance evaluation and testing); C4230M (Multiprocessor interconnection)

12/5/13 (Item 1 from file: 95)
DIALOG(R)File 95:TEME-Technology & Management

00555831 E92043423046
On mixing queries and transactions via multiversion locking
(Ueber das Mischen von Abfragen und Transaktionen durch Multiversionssperrung)
Bober, PM; Carey, MJ

Univ. of Wisconsin-Madison, USA
Computer Sciences Technical Report, University of Wisconsin, Computer Sciences Department, v531, nOkt, pp1-27, 1991
Document type: Report Language: English
Record type: Abstract

ABSTRACT:
In this paper, the authors discuss a new approach to multiversion **concurrency** control that allows high-performance transaction systems to support the on-line execution of long-running **queries** (e.g., for decision support purposes). Long-running **queries** are typically run at **level 1** or **level 2** consistency because they introduce a high level of data contention with two-phase locking. Multiversion algorithms have been discussed as a way to reduce the level of data contention and at the **same time** support the serializable execution of queries. The approach extends the multiversion locking algorithm developed by Computer Corporation of America by using record-level versioning and reserving a portion of each data page for **caching** prior versions that are potentially needed for the serializable execution of queries; on-page **caching** also enables an efficient approach to garbage collection of old versions. In addition, the authors introduce view sharing, which has the potential for reducing the cost of versioning by grouping together queries to run against the same transaction-consistent view of the database. Finally, they also present results from a simulation study that compares their approach versus that of providing level 1 or level 2 consistency for queries. The reults indicate that the authors approach is a viable alternative to reduced-consistency locking when the portion of each data reserved for prior versions is chosen appropriately.